# Destiliranje semantičke segmentacije

**Sučić, Filip**

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1658

# DESTILIRANJE SEMANTIČKE SEGMENTACIJE

Filip Sučić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1658

# DESTILIRANJE SEMANTIČKE SEGMENTACIJE

Filip Sučić

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# ZAVRŠNI ZADATAK br. 1658

| | |
|---|---|
| Pristupnik: | **Filip Sučić (0036540149)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentor: | prof. dr. sc. Siniša Šegvić |

Zadatak: **Destiliranje semantičke segmentacije**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim važnim primjenama. Jedna od velikih prepreka prema industrijskim izvedbama je velika računska složenost modela koji postižu najbolju točnost. Zbog toga su vrlo zanimljive metode za prijenos znanja prema modelima koji mogu zaključivati u stvarnom vremenu. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće segmentacijske arhitekture utemeljene na konvolucijama i pažnji. Odabrati brzi javno dostupni model te uhodati njegovo standardno učenje modela na potpuno označenim podatcima. Odabrati veliki javno dostupni model koji postiže izvrsnu generalizaciju te uhodati zaključivanje na javnim podatcima. Oblikovati destilacijsku proceduru za učenje brzog modela na pseudooznakama velikog modela. Validirati hiperparametre, vrednovati generalizacijsku izvedbu dvaju brzih modela te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate te potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 14. lipnja 2024.

*Zahvaljujem se mentoru prof. dr. sc. Siniši Šegviću i kolegi asistentu Ivanu Martinoviću na pomoći i strpljenju pri izradi ovog rada.*

# Sadržaj

# 1. Uvod

Računalni vid je područje umjetne inteligencije koje koristi metode strojnog učenja i neuronske mreže kako bi računala mogla izvlačiti korisne informacije iz slika, videozapisa i slično. Dok mi ljudi radimo to nesvjesno, takve procese teško je algoritamski opisati, što stvara veliki izazov u postizanju toga da računalo na neki način imitira naše sposobnosti.

U ovom radu najprije će se dati pregled neuronskih mreža općenito, kako funkcioniraju, uče i kako evaluiramo njihove rezultate. Zatim će fokus biti na konvolucijskim neuronskim mrežama, koje su dizajnirane za rad s vizualnim podacima. Bit će dan pregled konvolucijskih neuronskih mreža, njihove arhitekture i osnovna ideja kako funkcioniraju. U sklopu toga bit će objašnjeni pojmovi poput konvolucije, sažimanja i pažnje.

Jedan od zadataka računalnog vida je semantička segmentacija koja je centralni dio ovog rada. Unatoč velikom napretku u tom području, i dalje ostaje velik problem treniranja glomaznih modela koji postižu zapanjujuće rezulate, odnosno dobro generaliziraju. Zbog tog razloga razvila se ideja prenijeti to znanje s glomaznog modela na neki manji, koji će postizati solidne rezulate, ali biti efikasniji, tj. uz manje računalnih resursa, što je ključno za korištenje u stvarnom vremenu. Taj proces naziva se destilacija.

Bit će istražena destilacija znanja u kontekstu semantičke segmentacije na skupu podataka Cityscapes[2]. Korištena su dva modela: Swiftnet[3] kao model student i Mask2Former[4] kao model učitelj. Kroz eksperimente će se pokazati važnost odabira hiperparametara pri učenju modela te će se moći usporediti korištene metode. Također će se moći usporediti različite temperature u funkciji softmax, odnosno kako to utječe na treniranje i konačan rezultat. Na kraju će biti dana motivacija za daljnji rad i načini za poboljšanje dobivenih rezultata.

# 2. Duboko učenje

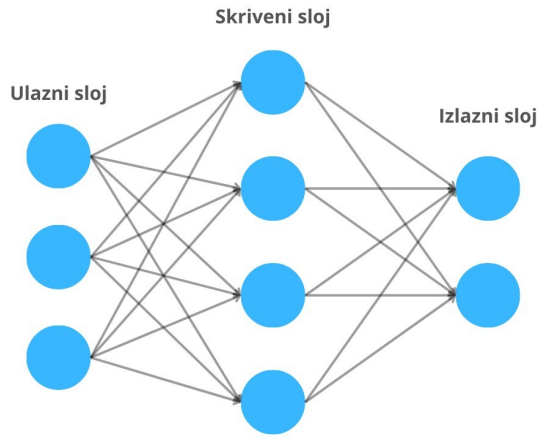
Duboko učenje je grana strojnog učenja koja je posebno prikladna za rješavanje problema iz područja umjetne inteligencije. Temelji se na predstavljanju podataka složenim reprezentacijama do kojih se dolazi slijedom naučenih nelinearnih transformacija. Neuronske mreže koje se koriste u dubokom učenju sastoje se od slojeva neurona koji su inspirirani strukturom ljudskog mozga. Ovi slojevi uključuju ulazni sloj, jedan ili više skrivenih slojeva, i izlazni sloj. Skriveni slojevi mogu imati različite oblike, kao što su konvolucijski slojevi, rekurentni slojevi, i potpuno povezani slojevi, svaki sa specifičnim funkcijama i primjenama. U ovom radu poseban fokus će biti na konvolucijskim slojevima te kako su oni ukomponirani u arhitekture konvolucijskih neuronskih mreža.

## 2.1. Umjetne neuronske mreže

Cilj umjetnih neuronskih mreža jest imitirati pravu (biološku) neuronsku mrežu. Naime, poznato je da se mozak sastoji od enormnog broja neurona koji svoje funkcije obavljaju paralelno. Zato gradimo umjetne neuronske mreže koje podatke obrađuju na jednak način. Umjetna neuronska mreža sastavljena je od velikog broja međusobno povezanih umjetnih neurona. Umjetni neuron je jednostavna procesna jedinica čija se funkcionalnost temelji na upravo biološkom neuronu. Kako bi umjetni neuron imitirao biološki, svaki ulaz u neuron $x_i$ množi se s osjetljivošću tog ulaza $w_i$ i to se sumira. Zatim se ukupnoj sumi dodaje pomak $b$. To se propušta kroz aktivacijsku funkciju $f$ čime nastaje izlazna vrijednost:

$$f(\sum_{i=1}^{n}(x_i \cdot w_i) + b) \tag{2.1}$$

Klasičan primjer umjetne neuronske mreže bi bila potpuno povezana umjetna neuronska mreža:




**Slika 2.1.** Primjer potpuno povezane neuronske mreže tipa 3x4x2

## 2.2. Aktivacijske funkcije

Prvi umjetni neuron bio je TLU-perceptron. On je kao aktivacijsku funkciju $f$ koristio funkciju skoka:

$$step(net) = \begin{cases} 0 & \text{ako } net < 0 \\ net & \text{ako } net \geq 0 \end{cases} \tag{2.2}$$

Još neke često korištene funkcije su sigmoidalna funkcija:

$$\sigma(net) = \frac{1}{1 + e^{-net}} \tag{2.3}$$

koja je prikladnija za korištenje kao aktivacijsku funkciju zbog svoje glatkoće i svojstva derivabilnosti nad cijelom domenom, no ona pak ima problem nestajućeg gradijenta: gradijent je za velike vrijednosti vrlo malen, što može drastično poremetiti učenje mreže. Taj problem rješava funkcija ReLU (zglobnica):

$$relu(net) = net^{+} = max(0, net) = \begin{cases} net & \text{ako } net > 0 \\ 0 & \text{inače} \end{cases} \tag{2.4}$$

Ona pak ima problem "umirućih" neurona; ako je ulaz negativan, aktivacija će biti nula pa ti neuroni ne prenose informacije kroz mrežu. Taj problem potencijalno se može riješiti pomoću propusne zglobnice:

$$f(net) = \begin{cases} net & \text{ako } net > 0 \\ \alpha \cdot net & \text{inače} \end{cases} \tag{2.5}$$

što omogućuje mali pozitivan gradijent kada je neuron neaktivan.[5]

## 2.3. Učenje neuronske mreže

Pri učenju neuronske mreže postoje 3 pristupa: nadzirano (učenje s učiteljem), nenadzirano (učenje bez učitelja) i podržano učenje. Kod nadziranog učenja podaci su zadani kao (ulaz, izlaz) = $(\mathbf{x}, y)$ i cilj učenja je naći preslikavanje $\hat{y} = f(\mathbf{x})$. Kod nenadziranog učenja dani su podaci bez ciljne vrijednosti, a cilj je naći pravilnost u ulaznim podacima: grupiranje, otkrivanje stršećih vrijednosti ili smanjiti dimenzionalnost. Kod podržanog učenja cilj je naučiti optimalnu strategiju na temelju pokušaja uz odgođenu nagradu. Naš fokus u ovom radu je na nadziranom učenju. Ako je $y$ diskretna ili nebrojčana varijabla, tada se radi o klasifikaciji, a kada je $y$ kontinuirana varijabla, radi se o regresiji[6] [7]. U ovom radu ćemo se fokusirati na klasifikaciju.

Znanje neuronske mreže zapravo je pohranjeno u težinama između neurona (i pomacima). Učenje neuronske mreže svodi se na pravilno podešavanje tih težina i pomaka. Neuronska mreža sa slike 2.1. ima tri sloja. Za svaki sloj trebamo znati matricu težina, vektor pomaka i aktivacijsku funkciju. Izlaz sloja $h_i$ možemo izraziti kao:

$$\vec{h_i} = f(\mathbf{W}_i \cdot \vec{h}_{i-1} + \vec{b}_i) \tag{2.6}$$

gdje je $f$ aktivacijska funkcija, $\mathbf{W}_i$ matrica težina i $\vec{b}_i$ vektor pomaka, uz $\vec{h_0} = \vec{x}$ i $\vec{y} = \vec{h}_{n-1}$ ako imamo n slojeva.

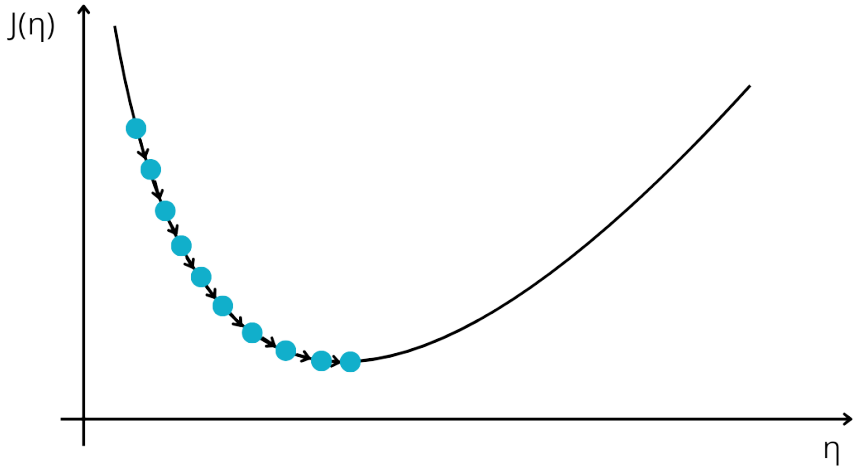
Da bismo uveli algoritme za prilagođavanje težina, odnosno algoritme učenja, najprije moramo definirati funkciju pogreške. Funkcija pogreške govori nam koliko su naše predikcije dobre i na koji način moramo promijeniti težine (i pomake). Pogreške se računaju na temelju izlaza modela i očekivanog izlaza. Često korištena funkcija pogreške (također ona koja je korištena i u ovom radu) je unakrsna entropija:

$$H(p,q) = -\sum_{x} p(x) log q(x) \tag{2.7}$$

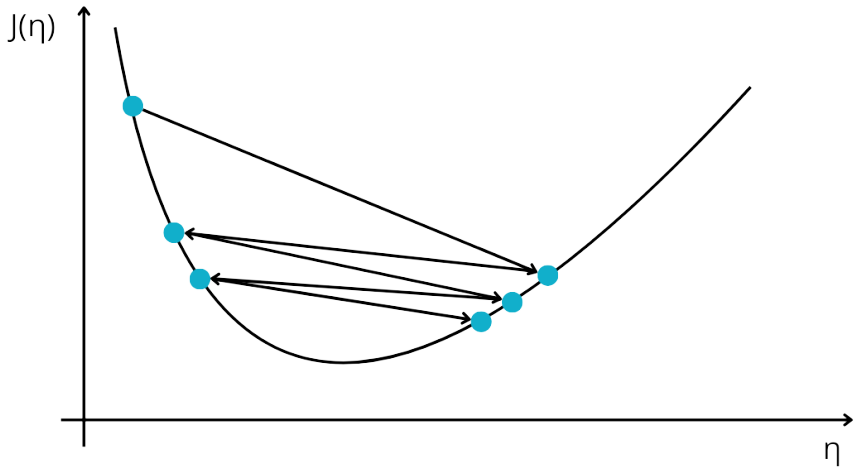
Jednom kada imamo definiranu funkciju pogreške, cilj nam je smanjiti ju. Ako je derivabilna, moguće je napisati optimizacijski postupak koji se temelji na izračunu gradijenata. Taj postupak naziva se postupak propagacije pogreške unatrag (eng. backpropagation) i on se najčešće koristi pri učenju neuronskih mreža.

$$w^l \leftarrow w^l - \eta \frac{\partial L}{\partial w^l} \tag{2.8}$$

Jednadžba 2.8 prikazuje izraz za ažuriranje težine $w$ u sloju $l$. $\eta$ je stopa učenja (eng. learning rate), a $\frac{\partial L}{\partial w^l}$ gradijent funkcije gubitka L u odnosu na težinu $w$. Kao što je i prikazano u ovom radu, $\eta$ mora biti pažljivo odabrana. Općenito je to mali pozitivan broj koji regulira u kojoj će se mjeri ažurirati trenutne vrijednosti težina. Problem leži u tome što ako je premali, model će presporo konvergirati, a ako je prevelik, može ne konvergirati uopće, tj. mreža ne uči.




**Slika 2.2.** Primjer premale stope učenja; prespora konvergencija




**Slika 2.3.** Primjer prevelika stope učenja; divergencija

Ključni koraci postupka propagacije pogreške unatrag su unaprijedni prolaz, prolaz unatrag i ažuriranje težina. Za ažuriranje težina bi se trebao koristiti cijeli skup podataka, no to je najčešće preskupo i nezgrapno. Umjesto toga, težine se ažuriraju nakon svakog uzorka ili nakon određenog podskupa podataka (tzv. minigrupa). Iterativni optimizacijski algoritam koji koristi propagaciju pogreške unatrag naziva se stohastički gradijentni spust (eng. Stochastic Gradient Descent, SGD). Stohastički je upravo zato što procjenuje gradijente na temelju jednog uzorka ili minigrupe.

Jednom kada imamo naučeni model, voljeli bismo izmjeriti njegove performanse. To se najčešće radi pomoću točnosti i matrice zabune. Točnost (eng. accuracy) je definirana kao omjer točno klasificiranih primjera i ukupnog broja primjera:

$$accuracy = \frac{correct}{total} \tag{2.9}$$

Točnost može biti problematična metrika zato što naš model može bolje funkcionirati za jednu klasu u odnosu na druge. Metrika koja je otporna na to je matrica zabune.

Kod binarne klasifikacije matrica zabune izgledala bi ovako:

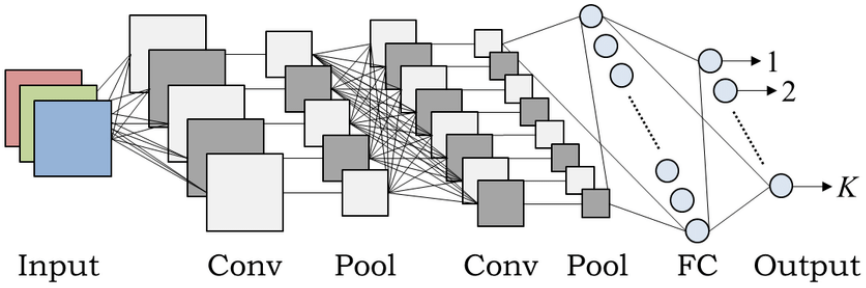
| | | Predicted class: Positive | Predicted class: Negative |
|---|---|---|---|
| Actual class | Positive | TP | FN |
| | Negative | FP | TN |

**Slika 2.4.** Primjer matrice zabune. Preuzeto s [8]

Generalno, dimenzije matrice zabune su $Y \times Y$ gdje je $Y$ broj klasa. Kod idealnog modela matrica zabune bila bi dijagonalna matrica. Koriste se metrike TP (točno pozitivno, eng. True Positive), TN (točno negativno, eng. True Negative), FP (netočno pozitivno, eng. False Positive) i FN (netočno negativno, eng. False Negative). TP je broj točno klasificiranih primjeraka promatrane klase, TN je broj točno klasificiranih primjeraka svih osim promatrane klase, FP je broj primjeraka koji ne pripadaju promatranoj klasi, ali su pogrešno klasificirani kao pripadnici te klase, a FN je broj primjeraka promatrane klase koji su pogrešno klasificirani kao ne pripadnici te klase.

## 2.4. Konvolucijske neuronske mreže

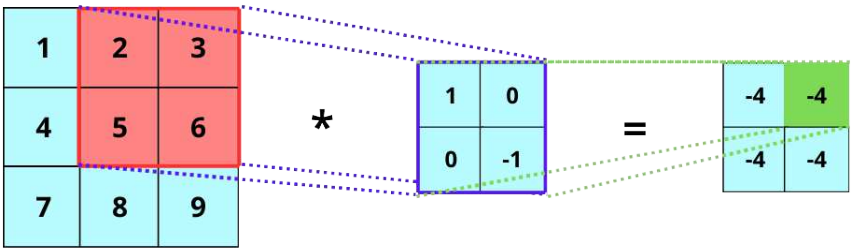
Konvolucijske neuronske mreže (eng. Convolutional Neural Networks, CNN) tip su neuronskih mreža koje se najčešće koriste u području računalnog vida. One, osim potpuno povezanih slojeva, imaju konvolucijske slojeve i slojeve sažimanja (eng. pooling).




**Slika 2.5.** Primjer arhitekture konvolucijske neuronske mreže. Preuzeto s [9]

### 2.4.1. Konvolucija

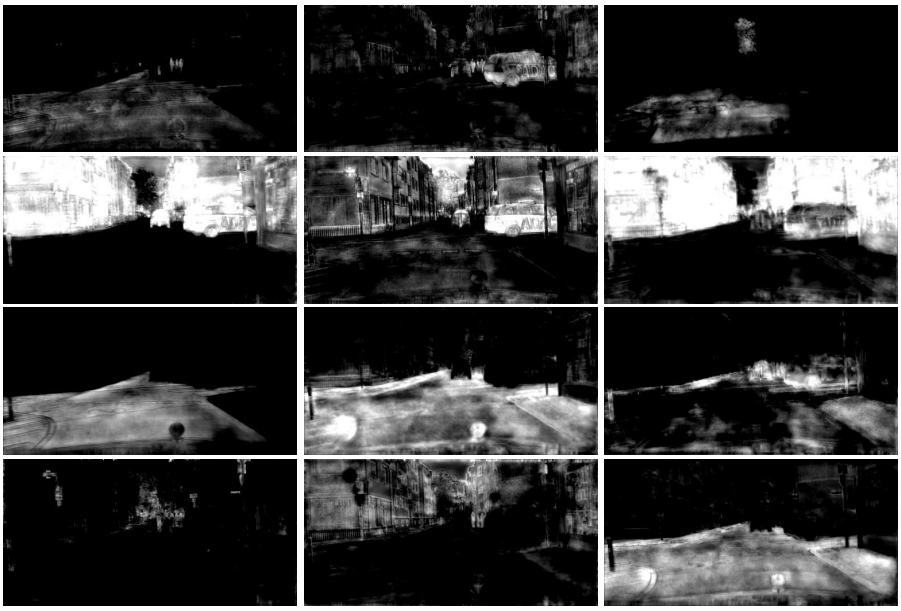
Formalna definicija konvolucije ovdje neće biti navedena; ukratko, to je matematička operacija koja djeluje na funkcije $f$ i $g$, a kao rezultat daje $f * g$. Koristi se za ekstrakciju značajki. Značajke u ovom kontekstu bili bi rubovi objekata, oblici i generalne strukture. Na taj način mreža uči obrasce koji su važni pri klasifikaciji. Pritom postoje različiti filtri (jezgre, eng. kernel) koji se koriste za različite ciljeve, npr. za detekciju rubova (vertikalnih, horizontalnih), zamagljivanje slike i sl. Ulazna matrica je slika; ako je jednokanalna onda je dvodimenzionalna, ako je RBG onda trodimenzionalna. Filter ili jezgra je manja matrica dimenzija $k \times k$. On se pomiče preko ulazne matrice i računa se produkt element-po-element (eng. element-wise) i zatim se to sumira. Kako bi se kontrolirale dimenzije izlazne matrice često se nadodaju nule oko ulazne matrice (eng. padding).




**Slika 2.6.** Primjer konvolucije 2x2. Objašenjenje rezultata: $2 \cdot 1 + 3 \cdot 0 + 5 \cdot 0 + 6 \cdot (-1) = -4$

Rezultat primjene konvolucijskog filtra naziva se mapa značajki.

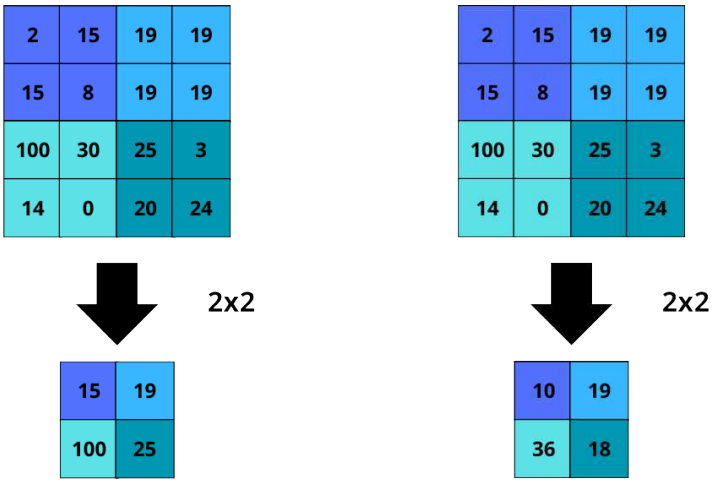
**Slika 2.7.** Primjer slike iz skupa podataka Cityscapes[2] za koju ćemo generirati značajke.



**Slika 2.8.** Razne značajke generirane pomoću modela ResNet-18[10] (temeljni model Swiftneta[3])

### 2.4.2. Sažimanje

Slojevi sažimanja u konvolucijskim neuronskim mrežama koriste se za smanjenje dimenzionalnosti značajki koje generiraju konvolucijski slojevi, ali tako da značajne informacije ostanu sačuvane. To se radi da bi se smanjila računalna kompleksnost. Definira se "prozor" dimenzija $k \times k$ koji pomičemo po značajkama s određenim korakom (eng. stride). Kod sažimanja maksimalnom vrijednošću (eng. max pooling) uzimamo maksimalnu vrijednost unutar prozora, a kod sažimanja srednjom vrijednošću (eng. average pooling) uzimamo aritmetičku sredinu vrijednosti unutar prozora.




**Slika 2.9.** Primjeri sažimanja. Lijeva slika prikazuje sažimanje maksimalnom vrijednošću, a desna sažimanje srednjom vrijednošću.

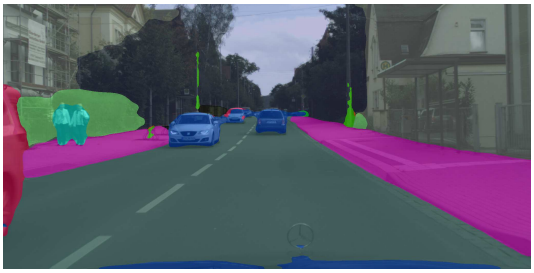
### 2.4.3. Pažnja

U okviru konvolucijskih arhitektura temeljenih na pažnji "pažnja" se odnosi na mehanizme koji omogućuju modelima da se fokusiraju na određene dijelove ulaznih podataka, a ostale dijelove zanemare. Inspirirana je načinom na koji i mi obrađujemo informacije poput teksta i slika. Čitajući rečenicu, ne pridajemo jednaku pozornost svim riječima, nego se fokusiramo na one koje su ključne za tu rečenicu, odnosno one koje nose najviše informacije. Pažnja se može implementirati na različite načine ovisno o potrebi i zadatku. Najčešće se to radi dodavanjem dodatnog sloja koji pridodaje težine mapi značajki.

# 3. Semantička segmentacija

Semantička segmentacija jedan je od tri tipa segmentacije slike u području računalnog vida. Cilj jest kategorizacija svakog od piksela slike u jednu od unaprijed zadanih klasa, npr. auto, drvo, prometni znak, cesta i sl. Semantička segmentacija se primjenjuje pri razvoju autonomnih vozila, u medicini pri analizi medicinskih slika, percepcije robota itd. Za semantičku segmentaciju najprikladnije su upravo konvolucijske neuronske mreže.

**Slika 3.1.** Primjer fotografije iz Cityscapes dataseta. Preuzeto s [2]



**Slika 3.2.** Semantička segmentacija. Generirano pomoću [4]

Osim semantičke, postoje i segmentacija instanci i panoptička segmentacija. Segmentacija instanci identificira i označava svaku pojedinu instancu objekta na slici, pritom dodjeljujući svakoj instanci jedinstveni identifikator.

**Slika 3.3.** Primjer fotografije iz Cityscapes dataseta. Preuzeto s [2]

**Slika 3.4.** Segmentacija instanci. Generirano pomoću [4]

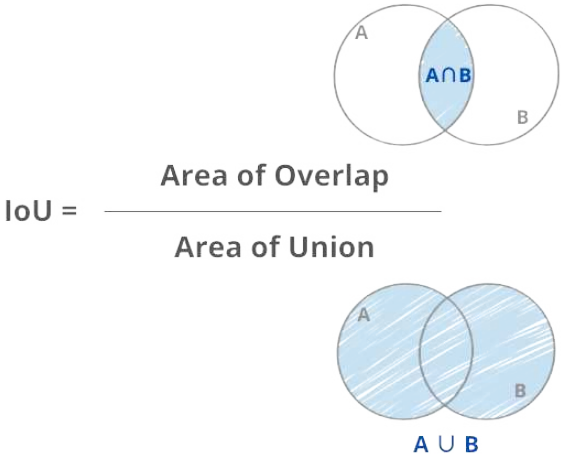
Panoptička segmentacija spoj je semantičke i segmentacije instanci. Ona svakom pikselu slike daje semantičko značenje, no osim toga razlikuje i različite instance istih klasa na slici.

**Slika 3.5.** Primjer fotografije iz Cityscapes dataseta. Preuzeto s [2]

**Slika 3.6.** Panoptička segmentacija. Generirano pomoću [4]

## 3.1. Mjera performansi

Najčešće korištena mjera performansi kod semantičke segmentacije, odnosno ona koja je korištena i u ovom radu je mIoU (eng. mean Intersection over Union). Općenito IoU, poznat kao i Jaccardov index, mjera je kvalitete preklapanja predviđene i stvarne segmentacije:




**Slika 3.7.** Vizualni prikaz kako se računa IoU

U ovom radu za IoU koristi se formula:

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.1}$$

MIoU je prosjek IoU vrijednosti za sve klase iz skupa podataka:

$$mIou = \frac{1}{K}\sum_{i=1}^{K} IoU_i \tag{3.2}$$

u slučaju K klasa. Osim mIoU koristi se i točnost klasifikacije na razini piksela (eng. pixel accuracy). Računa se pomoću formule 2.9

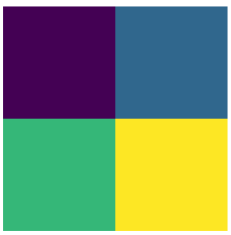
## 3.2. Interpolacija

Prilikom semantičke segmentacije slike, odnosno segmentacije općenito, slika se najprije obrađuje na određene načine: izrezuje, okreće i sl. Također se mogu i prilagoditi njene dimenzije. To se radi pomoću interpolacije.

Interpolacija je metoda procjene nepoznatih vrijednosti podataka iz već poznatih. U kontekstu obrade slika koristi se povećanje dimenzija i poboljšanje kvalitete slike. Najčešće se radi na jedan od tri načina:

1. metodom najbližeg susjeda
2. bilinearno
3. bikubično.

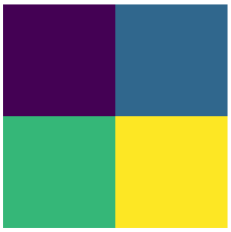
Na primjeru matrice 2x2 bit će prikazani navedene metode interpolacije.



**Slika 3.8.** Matrica koju ćemo interpolirati s veličine 2x2 na veličinu 16x16.

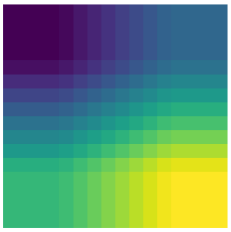
Metoda najbližeg susjeda najjednostavnija je od navedenih metoda interpolacije. Gleda samo najbližeg susjeda, odnosno ne uzima u obzir ostale poznate susjedne podatke.

Njeno djelovanje vrlo je intuitivno i može se vidjeti na sljedećoj slici:



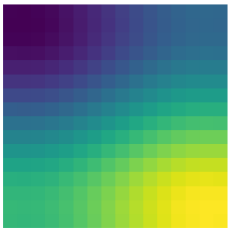
**Slika 3.9.** Rezultat interpolacije metodom najbližeg susjeda.

Bilinearna interpolacija nešto je složenija. Koristi linearne interpolacije prvo u jednom, a zatim u drugom smjeru (horizontalno i vertikalno) kako bi izračunala vrijednosti novih piksela. Prednost nad metodom najbližeg susjeda je ta da generira glatkije prijelaze, no problem je što može zamutiti oštre rubove.



**Slika 3.10.** Rezultat bilinearne interpolacije

Bikubična interpolacija koristi polinome trećeg reda za izračunavanje novih vrijednosti piksela. Ova metoda koristi 16 najbližih piksela (4x4 susjedstvo). Proizvodi glatke i precizne prijelaze, bolje očuvanje detalja u usporedbi s bilinearnom interpolacijom.



**Slika 3.11.** Rezultat bikubične interpolacije

Modeli korišteni u ovom radu koriste sve navedene metode interpolacije.

# 4. Destiliranje znanja u dubokom učenju

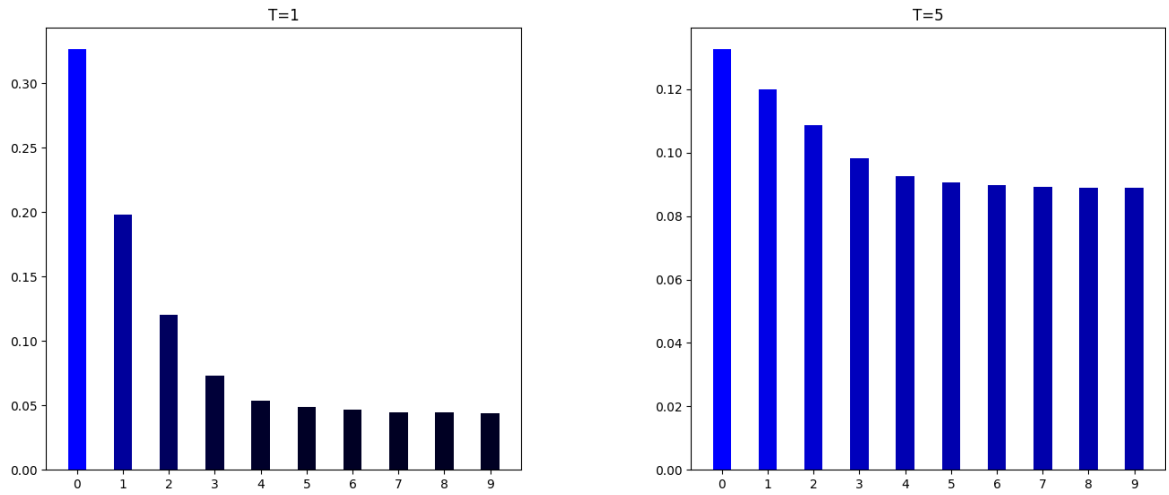
Najjednostavniji način za poboljšati performanse bilo kojeg algoritma strojnog učenja je naučiti puno različitih modela na istom skupu podataka i uzeti srednju vrijednost njihovih predikcija. Problem je što takav način rada može biti jako računski zahtjevan i nemoguć za praktično korištenje od strane velikog broja korisnika u stvarnom vremenu. Ideja destiliranja znanja u dubokom učenju bila bi prvo obaviti vremenski i računski zahtjevno treniranje velikog modela koji postiže odličnu generalizaciju, a zatim nekako prenijeti znanje tog modela na neki manji model koji će onda također postizati zadovoljavajuće rezultate i pritom biti korišten u stvarnom vremenu.

## 4.1. Tehnike destilacije

Kod klasifikacije, neuronske mreže na izlazu daju vjerojatnosti svake klase korištenjem funkcije softmax na logite:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4.1}$$

T je temperatura koja je najčešće postavljena na 1. Veća temperatura kao rezultat imat će mekšu distribuciju vjerojatnosti po klasama.




**Slika 4.1.** Efekt temperature kod softmaxa. Primjer s temperaturom 1 (lijevo) i 5 (desno)

Do tvrdih oznaka dolazimo tako da odaberemo klasu s najvećom vjerojatnošću. Konkretno, nakon što izračunamo vjerojatnosti klasa korištenjem softmax funkcije, za svaku instancu odabiremo klasu koja ima najveći $q_i$. Matematički, tvrda oznaka za instancu $x$ definirana je kao:

$$\hat{y} = \arg\max_i q_i \tag{4.2}$$

gdje je arg max funkcija definirana kao:

$$\arg\max_x f(x) := \{x \in X \,:\, f(s) \leq f(x) \text{ za sve } s \in X\}. \tag{4.3}$$

Meke oznake su interesantne jer preko njih imamo uvid u to koliko je model siguran u svoje odluke, no tvrde oznake su ipak ključne za donošenje krajnje odluke. Ako radimo prepoznavanje ručno napisanih znamenki, model će najčešće biti poprilično siguran u svoju odluku, no velik dio informacija o naučenom preslikavanju pohranjen je u omjerima mekih oznaka. Na primjer, ako je napisana znamenka jedan, pri klasifikaciji vjerojatnost da je to znamenka sedam može biti veća od ostalih, što nam govori da je model naučio da neke jedinice izgledaju kao sedmice. Ako koristimo destilaciju samo na temelju tvrdih oznaka, ta vrijedna informacija se gubi.

Najjednostavniji način destilacije bio bi istrenirati veliki model (učitelj) s visokom temperaturom u funkciji softmax i to koristiti kao ciljne meke oznake za mali model (student) koji koristi istu temperaturu. Nakon što je treniranje završilo, koristi se temperatura 1. Po uzoru na rad *Structured knowledge distillation for semantic segmentation*[1] koristila se i temperatura 1 pri treniranju te opisana tehnika destilacija na razini piksela (eng. pixel-wise distillation). U tom slučaju kao funkcija gubitka korištena je Kullback-Leibler divergencija:

$$D_{KL}(P||Q) = \sum_i P(i) \cdot log\frac{P(i)}{Q(i)} \tag{4.4}$$

gdje je P distribucija studenta, a Q distribucija učitelja. Kullback-Leibler divergencija često se naziva i relativna entropija zato što mjeri koliko se jedna distribucija razlikuje od druge (ali ne i koliko se druga razlikuje od prve, zato nije simetrična).

Ako imamo stvarne tvrde oznake za skup podataka za treniranje, tada možemo koristiti kombinaciju mekih učiteljevih oznaka i stvarnih tvrdih oznaka kao cilj pri učenju modela studenta. Koristi se ponderirana sredina dviju ciljnih funkcija. Prva ciljna funkcija je unakrsna entropija između mekih oznaka modela učitelja i mekih oznaka modela studenta koje su generirane s istom temperaturom kao i oznake učitelja. Druga ciljna funkcija je unakrsna entropija sa stvarnim tvrdim oznakama, pritom se tvrde oznake modela studenta generiraju s temperaturom 1. Preporučeno je koristiti znatno manju težinu za drugu ciljnu funkciju.[11]

U ovom radu također je obavljena destilacija samo pomoću tvrdih oznaka modela učitelja. Pritom je kao ciljna funkcija ponovno korištena unakrsna entropija. Osim pixel-wise destilacije, obavljena je i "image-wise" destilacija, odnosno u funkciji gubitka je uzeta Kullback-Leibler divergencija na razini svake slike u minigrupi.

# 5. Eksperimenti

## 5.1. Skup podataka

Za sve eksperimente korišten je skup podataka Cityscapes[12]. To je vrlo popularan skup podataka za segmentaciju u urbanoj okolini u području računalnog vida. Sastoji se od 30 klasa, od kojih je 19 klasa korišteno za evaluaciju performansi modela, podijeljenih u 8 grupa: ceste i pločnici, vozila, ljudi, zgrade, objekti (prometni znakovi, semafori), vegetacija, nebo i ostalo. Podijeljen je u tri podskupa: podskup za treniranje (2975 slika), podskup za validaciju (500 slika) i podskup za testiranje (1525 slika). Slike su fotografirane iz perspektive vozača koristeći kameru montiranu na vozilo.


**Slika 5.1.** Primjer fotografije iz Cityscapesa. Preuzeto s [2]

Podskup za treniranje nije javan (dostupne su samo slike, no ne i oznake), tako da će se rezultati odnositi na rezultate dobivene pri treniranju na podskupu za validaciju.

## 5.2. Korišteni modeli

Za potrebe ovog rada nisu osmišljeni novi modeli, nego su preuzeti već gotovi, s određenim izmjenama skripti za treniranje, odnosno generiranje izlaza. Modeli postižu sljedeće rezultate na skupu podataka Cityscapes:

| Classes | IoU - Mask2Former | IoU - Swiftnet |
|---|---|---|
| road | 0.984 | 0.975 |
| sidewalk | 0.870 | 0.816 |
| building | 0.930 | 0.914 |
| wall | 0.576 | 0.512 |
| fence | 0.628 | 0.556 |
| pole | 0.703 | 0.605 |
| traffic light | 0.753 | 0.653 |
| traffic sign | 0.825 | 0.757 |
| vegetation | 0.928 | 0.920 |
| terrain | 0.654 | 0.634 |
| sky | 0.951 | 0.943 |
| person | 0.844 | 0.799 |
| rider | 0.644 | 0.563 |
| car | 0.957 | 0.938 |
| truck | 0.685 | 0.632 |
| bus | 0.828 | 0.823 |
| train | 0.812 | 0.677 |
| motorcycle | 0.692 | 0.508 |
| bicycle | 0.801 | 0.751 |
| **mIoU** | **0.794** | **0.736** |

Tablica 5.1. Usporedba mIoU korištenih modela na Cityscapes skupu podataka

Pri destilaciji model učitelj će biti Mask2Former, a student Swiftnet.

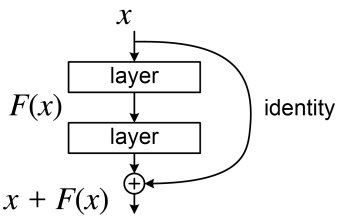
### 5.2.1. Swiftnet

Swiftnet je model razvijen na FER-u[3]. Sastoji se od tri osnovna građevna bloka: enkoder prepoznavanja, dekoder za povećanje rezolucije i modul za povećanje receptivnog polja.

Kao enkoder za segmentaciju korišten je model ResNet-18[10] predtreniran na ImageNetu[13]. ResNet-18 je pogodan za korištenje zbog relativno malog operativnog opterećenja, što omogućuje rad u stvarnom vremenu, umjerene dubine i rezidualne strukture. Rezidualna struktura znači da izlaz iz jednog sloja može preskočiti jedan ili više slojeva i biti direktno dodan izlazu nekog sloja dublje u mreži. Matematički se to može izraziti:

$$y = F(\mathbf{x}) + \mathbf{x} \tag{5.1}$$

kao što je i prikazano na 5.2.



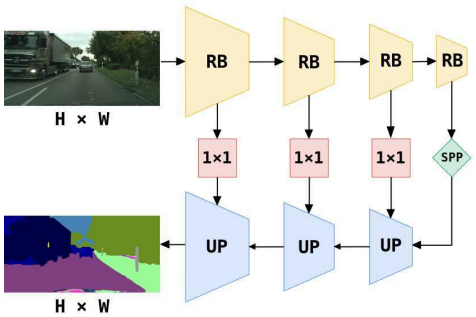

**Slika 5.2.** Rezidualni blok. Preuzeto s [14]

Imagenet je skup podataka koji sadrži više od 14 milijuna označenih slika raspoređenih u više od 20000 kategorija. Najčešće korišten podskup Imageneta (IMAGENET 1000) sadrži oko tisuću kategorija poput tarantula, puž, vagon itd.

**Slika 5.3.** Primjer slike iz skupa podataka Imagenet. Preuzeto s [14]

Dekoder je organiziran kao sekvenca modula za povećanje rezolucije s bočnim vezama. Moduli za povećanje rezolucije imaju dva ulaza: značajke niske rezolucije (koje treba povećati) i bočne značajke iz ranijeg sloja enkodera. Značajke niske rezolucije prvo se povećavaju bilinearnom interpolacijom na istu rezoluciju kao bočne značajke, a zatim se miješaju zbrajanjem i konačno obrađuju 3×3 konvolucijom.

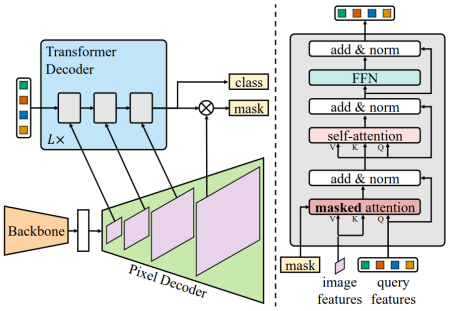
U ovom radu korišten je jednoskalni model:




**Slika 5.4.** Arhitektura jednoskalnog Swiftneta. Preuzeto s [3]

### 5.2.2. Mask2Former

Na slici 5.5. je prikazana generalna arhitektura modela Mask2Former. Taj model razvio je FacebookResearch tim (sadašnja Meta). Podržava univerzalnu segmentaciju slika (semantičku, segmentaciju instanci i panoptičku). Glavna inovacija ovog modela jest korištenje "maskirane pozornosti" (eng. masked attention). Kao enkoder (odnosno backbone) se koristi model ResNet-50 predtreniran na Imagenetu.




**Slika 5.5.** Arhitektura Mask2Formera. Preuzeto s [4]

## 5.3. Korištene tehnologije

### 5.3.1. PyTorch

PyTorch[15] je popularna biblioteka otvorenog koda koja pruža dvije bitne stvari: računanje s tenzorima i duboke neuronske mreže temeljene na sustavu automatske diferencijacije. Automatska diferencijacija (eng. autograd) je tehnika koja omogućava efikasno računanje gradijenata funkcije gubitka u odnosu na parametre mreže. Najčešće se koristi kao zamjena za NumPy[16] ili kao platforma za duboko učenje općenito. Snaga Pytorcha leži u tome što dopušta da tenzori s kojima se računa budu smješteni na GPU-u i tako ubrza račun.

### 5.3.2. Matplotlib

Matplotlib[17] je biblioteka otvorenog koda za statičke, dinamičke i interaktivne vizualizacije u Pythonu. Pruža jednostavne načine za razne grafove poput box-plotova, histograma, linijskih grafova itd. Vrlo je popularna zbog laganog korištenja uz druge popularne Python biblioteke poput NumPyja. U ovom radu korištena je za grafički prikaz performanski modela i grafova poput 4.1.

### 5.3.3. Google Colab

Google Colab[18] je besplatna platforma koja omogućuje pisanje i izvršavanje Python koda u internetskom pregledniku. Uz Colab dobiva se pristup snažnim GPU-ovima i TPU-ovima pa je posebno prikladan za duboko učenje i analizu podataka, pogotovo jer alati za to dolaze predinstalirani (PyTorch, TensorFlow, Keras itd.). GPU koji je korišten pri treniranju na Colabu je Nvidia L4. Colab pruža integraciju s Google Driveom što je posebno korisno pri učitavanju skupa podataka.

## 5.4. Rezultati

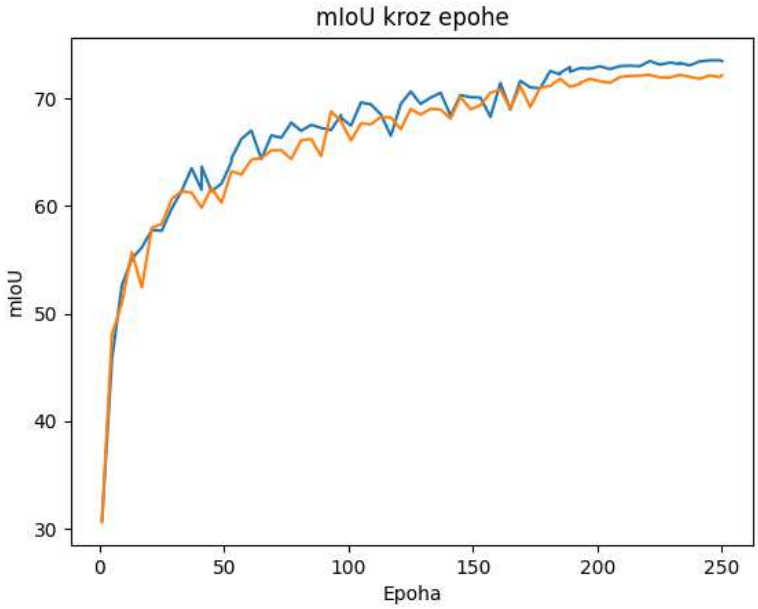
Eksperimenti su provedeni na pola rezolucije Cityscapesa. U eksperimentima je korišten CosineAnnealingLR iz Pytorcha. To je tehnika upravljanja brzinom učenja koja glatko smanjuje stopu učenja od $\eta_{max}$ do $\eta_{min}$ tijekom $T$ epoha. Za neke od eksperimenata mijenjane su vrijednosti minimalne stope učenja, odnosno $\eta_{min}$. Kao optimizator se koristio Adam. Slike su najprije nasumično okrenute, zatim je odrezan nasumični izrez slike veličine $448 \times 448$ piksela. Dio treniranja obavljen je na ZEMRIS-ovom udaljenom serveru, a dio na Colabu. GPU korišten na Colabu je kapacitetniji pa se nije koristila ista veličina minigrupe kao pri treniranju na serveru.

### 5.4.1. Prva metoda

Prva metoda odnosi se na destiliranje pomoću tvrdih oznaka Mask2Formera. Dobiven je krajnji rezultat od 72.20% mIou na skupu za validaciju. Treniranje je trajalo 250 epoha, veličina minigrupe bila je 14, stopa učenja $4 \cdot 10^{-4}$, a minimalna stopa učenja $10^{-6}$. Rezultati dobivenog modela usporedivi su s onima od modela treniranog na stvarnim oznakama. Formula funkcije gubitka glasi:

$$L = L_{CE}(\sigma(y), t) \tag{5.2}$$

gdje je $L_{CE}$ unakrsna entropija, $\sigma(y)$ su vjerojatnosti dobivene primjenom softmax funkcije na logite Swiftneta, a $t$ tvrde oznake Mask2Formera.

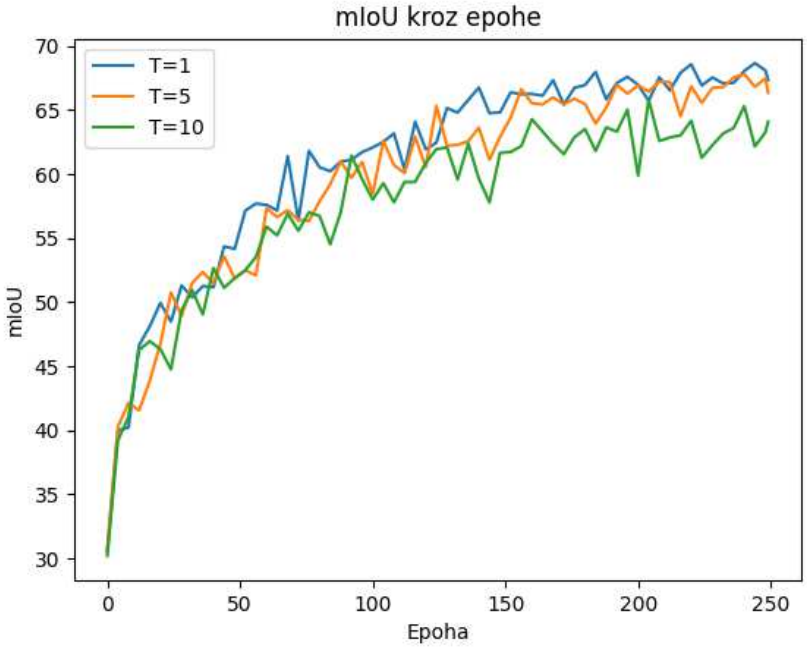



**Slika 5.6.** mIoU kroz epohe. Narančasta linija odgovara destiliranom modelu, a plava modelu treniranom na stvarnim oznakama.

### 5.4.2. Druga metoda

Druga metoda odnosi se na destiliranje pomoću mekih oznaka Mask2Formera. Obavljena je na dva načina: "image-wise" destilacija uz KL divergenciju kao funkciju gubitka i pixel-wise destilacija uz KL divergenciju kao funkciju gubitka.

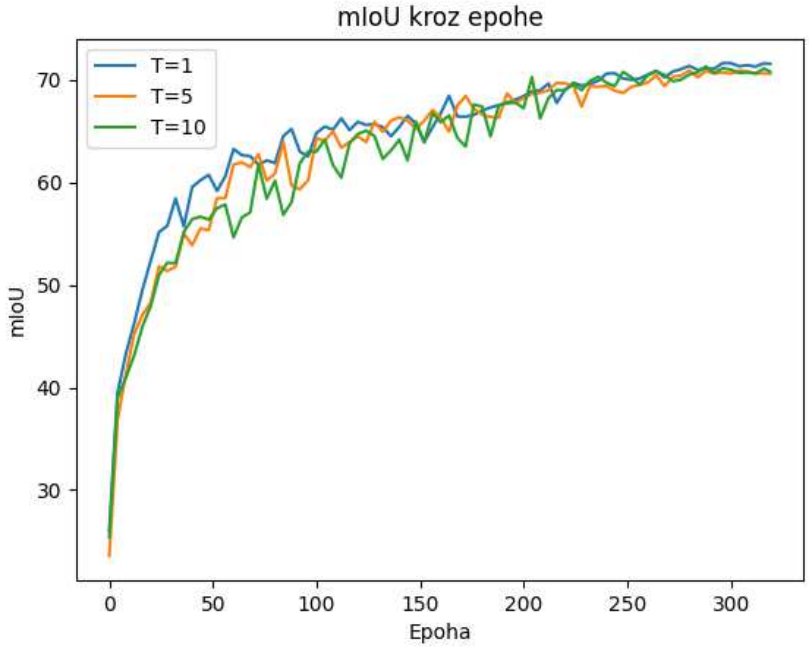
Što se tiče "image-wise" destilacije, treniranje je trajalo 250 epoha, veličina minigrupe bila je 3, stopa učenja $4 \cdot 10^{-4}$, a minimalna stopa učenja $10^{-6}$. Razlog za relativno malu minigrupu (u odnosu na treniranje na tvrdim oznakama) bila je veličina tenzora oznaka. Naime, najprije su generirane meke oznake Mask2Formera veličine $19 \times 256 \times 512$ kako bi se uštedjelo na memoriji. Zatim su pri treniranju oznake najprije bilinearno interpolirane na veličinu $19 \times 1024 \times 2048$. Nakon toga izrezuje se dio slike i oznaka veličine $448 \times 448$, odnosno $19 \times 448 \times 448$ piksela. Na taj način u funkciji gubitka računamo KL divergenciju između tenzora $3 \times 19 \times 448 \times 448$. Rezultat je tenzor iste veličine, sve vrijednosti tenzora se prosumiraju i podijele s veličinom minigrupe kako bi se dobio iznos funkcije gubitka. Tako je zadovoljena matematička definicija KL divergencije. Eksperiment je proveden uz temperature 1, 5 i 10.




**Slika 5.7.** mIoU kroz epohe. Plava linija odgovara temperaturi 1, narančasta temperaturi 5, a zelena temperaturi 10.

Razlog za ovako kaotično kretanje mIoU-a vjerojatno je prevelika stopa učenja i premala veličina minigrupe. Osim toga, sama ideja destilacije "image-wise" vjerojatno nije najmudrija. Najbolji rezultat dobiven je s temperaturom 1 (68.67% mIoU). Temperatura 10 je prevelika, gubi se previše informacija i zato je najgori rezultat dobiven s njom (65.81% mIoU). S temperaturom 5 dobiveni rezultat je 67.82% mIoU. Sve vrijednosti odnose se na podskup za validaciju. Ovaj eksperiment je bio neuspješan po rezultatima koje je dao, no ipak je bio koristan jer je uputio na to da treba smanjiti stopu učenja.

Nakon toga provedena je pixel-wise destilacija. U ovom eksperimentu sumirani tenzor veličine $3 \times 19 \times 448 \times 448$ koji dobijemo kao rezultat podijelimo s ukupnim brojem piksela, odnosno $3 \cdot 448 \cdot 448$. Za ove eksperimente je bilo potrebno smanjiti stopu učenja i minimalnu stopu učenja, a povećati broj epoha kako bi se dobili zadovoljavajući rezultati. Stopa učenja smanjena je na $10^{-4}$, a minimalna stopa učenja smanjena je na $10^{-7}$. Broj epoha povećan je na 320. Ovi eksperimenti provedeni su na Google Colabu[18] tako da je veličina grupa povećana na 6.




**Slika 5.8.** mIoU kroz epohe. Plava linija odgovara temperaturi 1, narančasta temperaturi 5, a zelena temperaturi 10.

Sve tri temperature su dale podjednak rezultat: 71.65% s temperaturom 1, 71.33% s temperaturom 10 i 70.94% s temperaturom 5.

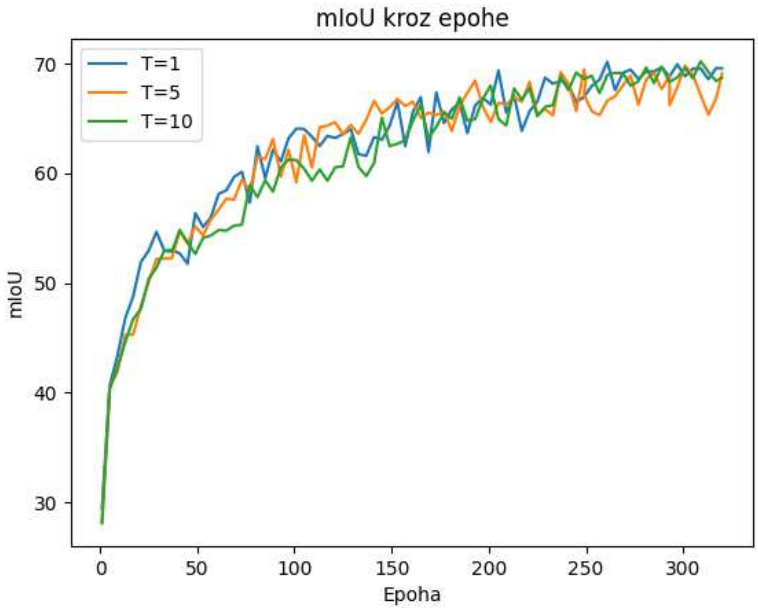
### 5.4.3. Treća metoda

Treća metoda se odnosi na destiliranje pomoću ponderirane sredine mekih oznaka Mask2Formera i stvarnih tvrdih oznaka. Eksperimenti su ponovno napravljeni s tri temperature: 1, 5 i 10. Stopa učenja i minimalna stopa učenja su za ove eksperimente smanjene na vrijednost $10^{-4}$, odnosno $10^{-7}$. Iz tog razloga broj epoha je povećan na 320. Veličina minigrupe je bila 3. Odabran je $\alpha = 0.8$, odnosno konačni izraz u funkciji gubitka glasi:

$$L = \alpha \cdot L_{CE}(\sigma(y), t) \cdot T^2 + (1 - \alpha) \cdot L_{CE}(\sigma(y), t_{true}) \tag{5.3}$$

gdje je $L_{CE}$ unakrsna entropija, $\sigma(y)$ vjerojatnosti dobivene primjenom softmax funkcije na logite Swiftneta, $t$ meke oznake modela učitelja, a $t_{true}$ stvarne tvrde oznake, kako je i opisano u radu *Distilling the Knowledge in a Neural Network*[11].

Sve temperature su radile podjednako dobro:




Postignuti su rezultati 70.17% (temperatura 1), 69.81% (temperatura 5) i 70.22% (temperatura 10).

### 5.4.4. Pregled rezultata

Konačan pregled svih rezultata sažeto je prikazan u 5.2.

| Metoda | Vel. minigrupe | LR | min LR | Broj epoha | Temperatura | mIoU |
|---|---|---|---|---|---|---|
| Stvarne oznake | 14 | 4e4 | 1e6 | 250 | 1 | 73.56 |
| Tvrde oznake | 14 | 4e4 | 1e6 | 250 | 1 | 72.20 |
| Pixel-wise | 6 | 1e4 | 1e7 | 320 | 1 | 71.65 |
| Pixel-wise | 6 | 1e4 | 1e7 | 320 | 10 | 71.33 |
| Pixel-wise | 6 | 1e4 | 1e7 | 320 | 5 | 70.94 |
| Treća ($\alpha = 0.8$) | 3 | 1e4 | 1e7 | 320 | 10 | 70.22 |
| Treća ($\alpha = 0.8$) | 3 | 1e4 | 1e7 | 320 | 1 | 70.17 |
| Treća ($\alpha = 0.8$) | 3 | 1e4 | 1e7 | 320 | 5 | 69.81 |
| "Image-wise" | 3 | 4e4 | 1e6 | 250 | 1 | 68.67 |
| "Image-wise" | 3 | 4e4 | 1e6 | 250 | 5 | 67.82 |
| "Image-wise" | 3 | 4e4 | 1e6 | 250 | 10 | 65.81 |

Tablica 5.2. Rezultati sortirani padajuće po mIoU.

Postignuti rezultati su vjerojatno mogli biti bolji uz bolju intuiciju o postavljanju prikladnijih vrijednosti hiperparametara i jačom računalnom snagom, ali su svakako zadovoljavajući.

# 6. Zaključak

Računalni vid i modeli dubokog učenja imaju široku primjenu u praksi, no često je ključno da se mogu koristiti u stvarnom vremenu. Zato se znanje većih modela može prenijeti na manje, efikasnije modele bez preznačajnog gubitka performansi, što je upravo i pokazano u ovom radu.

Dan je pregled ključnih koncepata dubokog učenja, opisane su neuronske mreže općenito i kako uče, s posebnim naglaskom na konvolucijske neuronske mreže i njihove primjene u semantičkoj segmentaciji. Opisani su korišteni modeli, osnove njihove arhitekture te skup podataka. Treniranje na stvarnim tvrdnim oznakama dalo je rezultat od 73.56% mIoU. Opisane su osnovne metode destiliranja znanja, uključujući destilaciju pomoću tvrdih oznaka, koja je dala rezultat od 72.2% mIoU. Ovaj rezultat direktno je usporediv s rezultatom treniranja na stvarnim tvrdim oznakama. Metoda destilacije pomoću mekih oznaka učitelja također je ispitana. Kod "image-wise" destilacije, rezultati su bili značajno lošiji nego kod ostalih metoda. S druge strane, pixel-wise destilacija dala je zadovoljavajuć rezultat od 71.65% mIoU, koji bi sigurno bio bolji uz pažljivije odabrane hiperparametre. Konačno, kombinacija stvarnih tvrdih oznaka i mekih oznaka učitelja dala je solidne rezultate od 70.22% mIoU.

Iako su korištene samo najosnovnije metode uz suboptimalne hiperparametre, i dalje su postignuti solidni rezultati što se čini obećavajuće za daljnji rad, koji bi uključivao složenije metode poput holističke ili "pair-wise" destilacije opisane u radu *Structured knowledge distillation for semantic segmentation*[1].

# Literatura


[1] Y. Liu, X. Zhang, Y. Hou, C. Wang, J. Feng, Y. Wei, i X. Xie, "Structured knowledge distillation for semantic segmentation", u *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019., str. 2604–2613.

[2] Cityscapes Dataset, "Cityscapes Dataset", https://www.cityscapes-dataset.com/downloads/, pristupljeno: 2024-06-06.

[3] M. Orsic, I. Kreso, P. Bevandić, i S. Segović, "In defense of pre-trained imagenet architectures for real-time semantic segmentation", u *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019., str. 12 607–12 616. https://doi.org/10.1109/CVPR.2019.01290

[4] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, i R. Girdhar, "Masked-attention mask transformer for universal image segmentation", 2022.

[5] A. L. Maas, A. Y. Hannun, i A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", u *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[6] B. Dalbelo Bašić i J. Šnajder, "Strojno učenje", 2020., Prezentacija dostupna u sklopu kolegija Uvod u umjetnu inteligenciju na FER-u.

[7] B. Dalbelo Bašić, M. Čupić, i J. Šnajder, "Neuronske mreže", 2020., Prezentacija dostupna u sklopu kolegija Uvod u umjetnu inteligenciju na FER-u.

[8] W. Commons, "Binary confusion matrix", 2021., pristupljeno: 2024-06-10. [Mrežno]. Adresa: https://commons.wikimedia.org/wiki/File:Binary_confusion_matrix.jpg

[9] T. Fernando, S. Sridharan, S. Denman, i C. Fookes, “Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks”, *ResearchGate*, 2017. [Mrežno]. Adresa: https://www.researchgate.net/publication/320748406_Consecutive_Dimensionality_Reduction_by_Canonical_Correlation_Analysis_for_Visualization_of_Convolutional_Neural_Networks

[10] K. He, X. Zhang, S. Ren, i J. Sun, “Deep residual learning for image recognition”, u *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016., str. 770–778. https://doi.org/10.1109/CVPR.2016.90

[11] G. Hinton, O. Vinyals, i J. Dean, “Distilling the knowledge in a neural network”, 2015.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, i B. Schiele, “The cityscapes dataset for semantic urban scene understanding”, u *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, i L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, u *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009., str. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[14] W. contributors, “Residual neural network – resblock”, 2023., pristupljeno: 2024-06-07. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Residual_neural_network#/media/File:ResBlock.png

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, i S. Chintala, “Pytorch: An imperative style, high-performance deep learning library”, 2019., arXiv:1912.01703. [Mrežno]. Adresa: http://arxiv.org/abs/1912.01703

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe,

P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, i T. E. Oliphant, “Array programming with numpy”, 2020., arXiv:2006.10256. [Mrežno]. Adresa: http://arxiv.org/abs/2006.10256

[17] J. D. Hunter, *Matplotlib: A 2D Graphics Environment*. John Wiley & Sons, 2007. [Mrežno]. Adresa: http://dx.doi.org/10.1109/MCSE.2007.55

[18] “Colaboratory”, https://research.google.com/colaboratory/faq.html, pristupljeno: 2024-06-10.

# Sažetak


## Destiliranje semantičke segmentacije

Filip Sučić

Problem kod modela dubokog učenja koji postižu zapanjujuće rezultate je što su često glomazni i ne možemo ih koristiti u stvarnom vremenu. U ovom radu se istražuju metode kako učinkovito prenijeti znanje takvih modela na manje, efikasnije modele. Prikazane su osnovne metode, uključujući destilaciju pomoću tvrdih i mekih oznaka, kao i kombinacija stvarnih tvrdih oznaka i mekih oznaka učitelja. Eksperimenti su provedeni na skupu podataka Cityscapes, pri čemu je model učitelj Mask2Former, a model student Swiftnet. Rezultati pokazuju da je moguće prenijeti znanje s većih modela na manje, uz zadržavanje zadovoljavajuće razine performansi.



**Ključne riječi:** računalni vid; duboko učenje; semantička segmentacija; tvrde oznake; meke oznake; destilacija znanja

# Abstract


## Destilling semantic segmentation

Filip Sučić

The issue with deep learning models that achieve outstanding results is that they are often too cumbersome to be used in real-time applications. This paper explores methods to effectively transfer the knowledge of such models to smaller, more efficient ones. Basic methods are presented, including distillation using hard and soft labels, as well as a combination of true hard labels and soft teacher labels. Experiments were conducted on the Cityscapes dataset, with Mask2Former as the teacher model and Swiftnet as the student model. The results demonstrate that it is possible to transfer knowledge from larger models to smaller ones while maintaining a satisfactory level of performance.



**Keywords:** computer vision; deep learning; semantic segmentation; hard labels; soft labels; knowledge distillation